LLM Fine-tuning Challenge: Enhancing Qwen 2.5 3B for AI Research QA

Team Sidemen

The quantized fine tuned model and other parameters can be found in this Hugginface Repository

Approach Overview

Here we fine-tune the **Qwen 2.5 3B-Instruct** model for answering technical AI research questions efficiently while optimizing for low-resource deployment through 4-bit quantization.

The process was divided into three key stages:

1. **Data Preparation:**
    - AI research papers were scraped from arXiv focusing on reasoning capabilities, reinforcement learning, and LLM optimization.
    - Text was extracted from PDFs and Markdown files, cleaned, and split into 4,000-character chunks with 500-character overlap to maintain context continuity.
2. **Synthetic QA Generation:**
    - GPT-4o was used to generate 2,000 high-quality QA pairs. The generated questions emphasized conceptual understanding, experimental results, and technical comparisons.
    - RAGAS filtering was applied to retain high-relevance responses (≥0.8 score). Manual audits confirmed technical accuracy.
3. **Fine-Tuning & Quantization:**
    - QLoRA (Quantized LoRA) was used for efficient fine-tuning, reducing VRAM consumption while maintaining accuracy.
    - The fine-tuned model was converted into 4-bit GGUF format using the `q4_k_m` quantization method.

Key Technical Choices

- **Why Qwen 2.5 3B-Instruct?**
    - Pre-trained on instruction-following tasks, making it ideal for fine-tuning in technical Q&A settings.
    - The 3B parameter size provides a balance between accuracy and hardware constraints.

- **QLoRA Configuration:**
  LoRA rank (`r=16`) and alpha (`α=32`) were chosen after testing different values for optimal memory efficiency.
    - Targeted attention layers: **`q_proj`**, **`v_proj`**, **`gate_proj`**, ensuring adaptation of key parameters without excessive memory overhead.

- **Synthetic Data Strategy:**
  - GPT-4o was selected over GPT-3.5 for generating deeper technical questions.

- **Evaluation Framework:**
  - The dataset was scored using RAGAS, prioritizing answer relevancy and faithfulness to prevent hallucinations.
  - A manual review of 5% of the dataset ensured alignment with research standards.

Dataset Preparation & QA Generation

Dataset Sourcing & Processing

- **Sources:** 20 AI research papers were scraped from arXiv, along with Markdown files containing AI-related technical content.
- **Chunking Strategy:**
  - Texts were split into overlapping segments (4,000 chars, 500 overlap) to ensure context retention.
  - Multiple Q&A pairs per chunk were generated to maximize information extraction.

```python
chunk_size = 4000
overlap = 500

chunks = [text[i:i+chunk_size] for text in all_text for i in range(0, len(text), chunk_size - overlap)]
```

Synthetic Q&A Generation & Filtering

- **Phase 1: QA Generation**
  - GPT-4o was prompted to generate detailed, structured question-answer pairs.
- **Phase 2: Data Filtering**
  - RAGAS filtering removed low-quality pairs based on answer relevancy (<0.8 score).
  - Short/incoherent responses were eliminated.
  - Duplicates were detected and removed using substring matching.

```python
df = df[df["answer_relevancy"] >= 0.8]  # Remove low-scoring Q&A pairs

df.drop_duplicates(subset=["question"], inplace=True)  # Remove duplicates
```

Training Process & Hyperparameters

Fine-Tuning Strategy

- **Low-Rank Adaptation (LoRA) + QLoRA**
  - Trained only adapter layers instead of full model fine-tuning.
  - VRAM Usage reduced allowing training on NVIDIA T4 GPUs.
  - Following configurations were used.

```python
model = FastLanguageModel.get_peft_model(
      model,
      r = 16,
      target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
            "gate_proj", "up_proj", "down_proj",],
      lora_alpha = 32,
      lora_dropout = 0.05,
      bias = "none",
      use_gradient_checkpointing = "unsloth",
      random_state = 3407,
      use_rslora = True,
      loftq_config = None
)
```

- **Training Hyperparameters:**

```python
training_args = TrainingArguments(
      output_dir="./qwen-finetuned",
      per_device_train_batch_size=4,
      gradient_accumulation_steps=4,
      warmup_steps=50,
      num_train_epochs=3,
      learning_rate=2e-4,
      fp16=True,
      logging_steps=10,
      optim="adamw_8bit",
      weight_decay=0.01,
      lr_scheduler_type="cosine",
      seed=42,
      save_strategy="epoch",
      report_to="none"
)
```

Deliverables & Submission Components

1. **Fine-Tuned Model:**
    - **Format:** `qwen-finetuned.gguf` (4-bit, 1.8 GB).
    - **Compatible with:** `llama.cpp`.
2. **Training Documentation:**
    - Includes hyperparameters, LoRA configurations, and dataset statistics.
3. **Evaluation Data:**
    - `filtered_score_df2.csv` contains high-quality QA pairs with RAGAS validation scores.

```python
model.save_pretrained_gguf("qwen-finetuned.gguf", tokenizer,
quantization_method="q4_k_m")
```

Challenges Faced

- **Quantization Instability:**
    - Initial attempts with `q4_0` led to output truncation. Switching to `q4_k_m` resolved this.
- **Dataset Quality Variability:**
    - Some research papers contained ambiguous text, requiring manual filtering to ensure answer quality.

Conclusion

This project successfully fine-tuned Qwen 2.5-3B-Instruct for AI research Q&A using QLoRA & 4-bit quantization.